Fibonacci:

```
@  ----------

@ Código de teste

@  ----------



func void fib(int n):
        [0005, 0001] (0015, FUNC      ) {func}
        [0005, 0006] (0041, VOID      ) {void}
        [0005, 0011] (0002, ID        ) {fib}
        [0005, 0014] (0010, BEGIN_PARM) {(}
        [0005, 0015] (0003, INT       ) {int}
        [0005, 0019] (0002, ID        ) {n}
        [0005, 0020] (0011, END_PARM  ) {)}
        [0005, 0021] (0008, BEGIN_SCP ) {:}

    int a = 0, b = 1, i = 0, aux
        [0006, 0005] (0003, INT       ) {int}
        [0006, 0009] (0002, ID        ) {a}
        [0006, 0011] (0030, OP_ATRIB  ) {=}
        [0006, 0013] (0017, CONST_INT ) {0}
        [0006, 0014] (0016, SEPARATOR ) {,}
        [0006, 0016] (0002, ID        ) {b}
        [0006, 0018] (0030, OP_ATRIB  ) {=}
        [0006, 0020] (0017, CONST_INT ) {1}
        [0006, 0021] (0016, SEPARATOR ) {,}
        [0006, 0023] (0002, ID        ) {i}
        [0006, 0025] (0030, OP_ATRIB  ) {=}
        [0006, 0027] (0017, CONST_INT ) {0}
        [0006, 0028] (0016, SEPARATOR ) {,}
        [0006, 0030] (0002, ID        ) {aux}

    while (a + b) < n:
        [0007, 0005] (0028, INS_WHILE ) {while}
        [0007, 0011] (0010, BEGIN_PARM) {(}
        [0007, 0012] (0002, ID        ) {a}
        [0007, 0014] (0034, OP_ADD    ) {+}
```

```
[0007, 0016] (0002, ID        ) {b}
[0007, 0017] (0011, END_PARM  ) {)}
[0007, 0019] (0037, OP_RELAT  ) {<}
[0007, 0021] (0002, ID        ) {n}
[0007, 0022] (0008, BEGIN_SCP ) {:}

if i > 0:
[0008, 0009] (0024, INS_IF    ) {if}
[0008, 0012] (0002, ID        ) {i}
[0008, 0014] (0037, OP_RELAT  ) {>}
[0008, 0016] (0017, CONST_INT ) {0}
[0008, 0017] (0008, BEGIN_SCP ) {:}

   show(",")
[0009, 0013] (0023, INS_SHOW  ) {show}
[0009, 0017] (0010, BEGIN_PARM) {(}
[0009, 0018] (0020, CONST_STR ) {","}
[0009, 0021] (0011, END_PARM  ) {)}

end
[0010, 0009] (0009, END_SCP   ) {end}

if i == 1:
[0011, 0009] (0024, INS_IF    ) {if}
[0011, 0012] (0002, ID        ) {i}
[0011, 0014] (0038, OP_REL_EQ ) {==}
[0011, 0017] (0017, CONST_INT ) {1}
[0011, 0018] (0008, BEGIN_SCP ) {:}

   show("0")
[0012, 0013] (0023, INS_SHOW  ) {show}
[0012, 0017] (0010, BEGIN_PARM) {(}
[0012, 0018] (0020, CONST_STR ) {"0"}
[0012, 0021] (0011, END_PARM  ) {)}

end
[0013, 0009] (0009, END_SCP   ) {end}

if i == 1:
[0014, 0009] (0024, INS_IF    ) {if}
[0014, 0012] (0002, ID        ) {i}
[0014, 0014] (0038, OP_REL_EQ ) {==}
[0014, 0017] (0017, CONST_INT ) {1}
```

```
[0014, 0018] (0008, BEGIN_SCP ) {:}

   show("1")
[0015, 0013] (0023, INS_SHOW  ) {show}
[0015, 0017] (0010, BEGIN_PARM) {(}
[0015, 0018] (0020, CONST_STR ) {"1"}
[0015, 0021] (0011, END_PARM  ) {)}

end
[0016, 0009] (0009, END_SCP   ) {end}

else:
[0017, 0009] (0026, INS_ELSE  ) {else}
[0017, 0013] (0008, BEGIN_SCP ) {:}

   aux = a + b
[0018, 0013] (0002, ID        ) {aux}
[0018, 0017] (0030, OP_ATRIB  ) {=}
[0018, 0019] (0002, ID        ) {a}
[0018, 0021] (0034, OP_ADD    ) {+}
[0018, 0023] (0002, ID        ) {b}

   show(aux)
[0019, 0013] (0023, INS_SHOW  ) {show}
[0019, 0017] (0010, BEGIN_PARM) {(}
[0019, 0018] (0002, ID        ) {aux}
[0019, 0021] (0011, END_PARM  ) {)}

   a = b
[0020, 0013] (0002, ID        ) {a}
[0020, 0015] (0030, OP_ATRIB  ) {=}
[0020, 0017] (0002, ID        ) {b}

   b = a + b
[0021, 0013] (0002, ID        ) {b}
[0021, 0015] (0030, OP_ATRIB  ) {=}
[0021, 0017] (0002, ID        ) {a}
[0021, 0019] (0034, OP_ADD    ) {+}
[0021, 0021] (0002, ID        ) {b}

end
[0022, 0009] (0009, END_SCP   ) {end}
```

```
    i = 1 + i
        [0023, 0005] (0002, ID        ) {i}
        [0023, 0007] (0030, OP_ATRIB  ) {=}
        [0023, 0009] (0017, CONST_INT ) {1}
        [0023, 0011] (0034, OP_ADD    ) {+}
        [0023, 0013] (0002, ID        ) {i}

    end
        [0024, 0005] (0009, END_SCP   ) {end}

end
        [0025, 0001] (0009, END_SCP   ) {end}




func void main():
        [0028, 0001] (0015, FUNC      ) {func}
        [0028, 0006] (0041, VOID      ) {void}
        [0028, 0011] (0001, MAIN      ) {main}
        [0028, 0015] (0010, BEGIN_PARM) {(}
        [0028, 0016] (0011, END_PARM  ) {)}
        [0028, 0017] (0008, BEGIN_SCP ) {:}

    int n
        [0029, 0005] (0003, INT       ) {int}
        [0029, 0009] (0002, ID        ) {n}

    input(n)
        [0030, 0005] (0022, INS_INPUT ) {input}
        [0030, 0010] (0010, BEGIN_PARM) {(}
        [0030, 0011] (0002, ID        ) {n}
        [0030, 0012] (0011, END_PARM  ) {)}

    fib(n)
        [0031, 0005] (0002, ID        ) {fib}
        [0031, 0008] (0010, BEGIN_PARM) {(}
        [0031, 0009] (0002, ID        ) {n}
        [0031, 0010] (0011, END_PARM  ) {)}

end
        [0032, 0001] (0009, END_SCP   ) {end}
```

Shellsort:

```
@ ---------------

@ Código de teste

@ ---------------



func void shellsort(int[] arr, int n):
        [0005, 0001] (0015, FUNC      ) {func}
        [0005, 0006] (0041, VOID      ) {void}
        [0005, 0011] (0002, ID        ) {shellsort}
        [0005, 0020] (0010, BEGIN_PARM) {(}
        [0005, 0021] (0003, INT       ) {int}
        [0005, 0024] (0012, BEGIN_ARR ) {[}
        [0005, 0025] (0013, END_ARR   ) {]}
        [0005, 0027] (0002, ID        ) {arr}
        [0005, 0030] (0016, SEPARATOR ) {,}
        [0005, 0032] (0003, INT       ) {int}
        [0005, 0036] (0002, ID        ) {n}
        [0005, 0037] (0011, END_PARM  ) {)}
        [0005, 0038] (0008, BEGIN_SCP ) {:}



    int i, j, t, temp
        [0007, 0005] (0003, INT       ) {int}
        [0007, 0009] (0002, ID        ) {i}
        [0007, 0010] (0016, SEPARATOR ) {,}
        [0007, 0012] (0002, ID        ) {j}
        [0007, 0013] (0016, SEPARATOR ) {,}
        [0007, 0015] (0002, ID        ) {t}
        [0007, 0016] (0016, SEPARATOR ) {,}
        [0007, 0018] (0002, ID        ) {temp}

    int = (n/2)
        [0008, 0005] (0003, INT       ) {int}
        [0008, 0009] (0030, OP_ATRIB  ) {=}
        [0008, 0011] (0010, BEGIN_PARM) {(}
```

```
    [0008, 0012] (0002, ID        ) {n}
    [0008, 0013] (0035, OP_MULTI  ) {/}
    [0008, 0014] (0017, CONST_INT ) {2}
    [0008, 0015] (0011, END_PARM  ) {)}

while i > 0:
    [0009, 0005] (0028, INS_WHILE ) {while}
    [0009, 0011] (0002, ID        ) {i}
    [0009, 0013] (0037, OP_RELAT  ) {>}
    [0009, 0015] (0017, CONST_INT ) {0}
    [0009, 0016] (0008, BEGIN_SCP ) {:}

    for j - i, n - 1,:
    [0010, 0009] (0027, INS_FOR   ) {for}
    [0010, 0013] (0002, ID        ) {j}
    [0010, 0015] (0040, OP_UNNEG  ) {-}
    [0010, 0017] (0002, ID        ) {i}
    [0010, 0018] (0016, SEPARATOR ) {,}
    [0010, 0020] (0002, ID        ) {n}
    [0010, 0022] (0040, OP_UNNEG  ) {-}
    [0010, 0024] (0017, CONST_INT ) {1}
    [0010, 0025] (0016, SEPARATOR ) {,}
    [0010, 0026] (0008, BEGIN_SCP ) {:}

        temp = arr[j]
    [0011, 0013] (0002, ID        ) {temp}
    [0011, 0018] (0030, OP_ATRIB  ) {=}
    [0011, 0020] (0002, ID        ) {arr}
    [0011, 0023] (0012, BEGIN_ARR ) {[}
    [0011, 0024] (0002, ID        ) {j}
    [0011, 0025] (0013, END_ARR   ) {]}

        while t >= i and (arr[t - i] > temp):
    [0012, 0013] (0028, INS_WHILE ) {while}
    [0012, 0019] (0002, ID        ) {t}
    [0012, 0021] (0037, OP_RELAT  ) {>=}
    [0012, 0024] (0002, ID        ) {i}
    [0012, 0026] (0031, OP_AND    ) {and}
    [0012, 0030] (0010, BEGIN_PARM) {(}
    [0012, 0031] (0002, ID        ) {arr}
    [0012, 0034] (0012, BEGIN_ARR ) {[}
    [0012, 0035] (0002, ID        ) {t}
    [0012, 0037] (0040, OP_UNNEG  ) {-}
```

```
[0012, 0039] (0002, ID        ) {i}
[0012, 0040] (0013, END_ARR   ) {]}
[0012, 0042] (0037, OP_RELAT  ) {>}
[0012, 0044] (0002, ID        ) {temp}
[0012, 0048] (0011, END_PARM  ) {)}
[0012, 0049] (0008, BEGIN_SCP ) {:}

        arr[t] - arr[t - 1]
[0013, 0017] (0002, ID        ) {arr}
[0013, 0020] (0012, BEGIN_ARR ) {[}
[0013, 0021] (0002, ID        ) {t}
[0013, 0022] (0013, END_ARR   ) {]}
[0013, 0024] (0040, OP_UNNEG  ) {-}
[0013, 0026] (0002, ID        ) {arr}
[0013, 0029] (0012, BEGIN_ARR ) {[}
[0013, 0030] (0002, ID        ) {t}
[0013, 0032] (0040, OP_UNNEG  ) {-}
[0013, 0034] (0017, CONST_INT ) {1}
[0013, 0035] (0013, END_ARR   ) {]}

        t = t - i
[0014, 0017] (0002, ID        ) {t}
[0014, 0019] (0030, OP_ATRIB  ) {=}
[0014, 0021] (0002, ID        ) {t}
[0014, 0023] (0040, OP_UNNEG  ) {-}
[0014, 0025] (0002, ID        ) {i}

    end
[0015, 0013] (0009, END_SCP   ) {end}

    arr[t] = temp
[0016, 0013] (0002, ID        ) {arr}
[0016, 0016] (0012, BEGIN_ARR ) {[}
[0016, 0017] (0002, ID        ) {t}
[0016, 0018] (0013, END_ARR   ) {]}
[0016, 0020] (0030, OP_ATRIB  ) {=}
[0016, 0022] (0002, ID        ) {temp}

end
[0017, 0009] (0009, END_SCP   ) {end}

i = (i/2)
[0018, 0009] (0002, ID        ) {i}
```

```
        [0018, 0011] (0030, OP_ATRIB  ) {=}
        [0018, 0013] (0010, BEGIN_PARM) {(}
        [0018, 0014] (0002, ID        ) {i}
        [0018, 0015] (0035, OP_MULTI  ) {/}
        [0018, 0016] (0017, CONST_INT ) {2}
        [0018, 0017] (0011, END_PARM  ) {)}

    end
        [0019, 0005] (0009, END_SCP   ) {end}

end
        [0020, 0001] (0009, END_SCP   ) {end}



func void main():
        [0022, 0001] (0015, FUNC      ) {func}
        [0022, 0006] (0041, VOID      ) {void}
        [0022, 0011] (0001, MAIN      ) {main}
        [0022, 0015] (0010, BEGIN_PARM) {(}
        [0022, 0016] (0011, END_PARM  ) {)}
        [0022, 0017] (0008, BEGIN_SCP ) {:}

    int size
        [0023, 0005] (0003, INT       ) {int}
        [0023, 0009] (0002, ID        ) {size}

    int[300] arr
        [0024, 0005] (0003, INT       ) {int}
        [0024, 0008] (0012, BEGIN_ARR ) {[}
        [0024, 0009] (0017, CONST_INT ) {300}
        [0024, 0012] (0013, END_ARR   ) {]}
        [0024, 0014] (0002, ID        ) {arr}

    int i
        [0025, 0005] (0003, INT       ) {int}
        [0025, 0009] (0002, ID        ) {i}

    show("Digite o tamanho da sequencia (limite de 300)"
        [0026, 0005] (0023, INS_SHOW  ) {show}
        [0026, 0009] (0010, BEGIN_PARM) {(}
        [0026, 0010] (0020, CONST_STR ) {"Digite o tamanho da sequencia (limite de 300)"}
```

```
input(size)
    [0027, 0005] (0022, INS_INPUT ) {input}
    [0027, 0010] (0010, BEGIN_PARM) {(}
    [0027, 0011] (0002, ID        ) {size}
    [0027, 0015] (0011, END_PARM  ) {)}

for i - 0, size - 1, :
    [0028, 0005] (0027, INS_FOR   ) {for}
    [0028, 0009] (0002, ID        ) {i}
    [0028, 0011] (0040, OP_UNNEG  ) {-}
    [0028, 0013] (0017, CONST_INT ) {0}
    [0028, 0014] (0016, SEPARATOR ) {,}
    [0028, 0016] (0002, ID        ) {size}
    [0028, 0021] (0040, OP_UNNEG  ) {-}
    [0028, 0023] (0017, CONST_INT ) {1}
    [0028, 0024] (0016, SEPARATOR ) {,}
    [0028, 0026] (0008, BEGIN_SCP ) {:}

    input(arr[i])
    [0029, 0009] (0022, INS_INPUT ) {input}
    [0029, 0014] (0010, BEGIN_PARM) {(}
    [0029, 0015] (0002, ID        ) {arr}
    [0029, 0018] (0012, BEGIN_ARR ) {[}
    [0029, 0019] (0002, ID        ) {i}
    [0029, 0020] (0013, END_ARR   ) {]}
    [0029, 0021] (0011, END_PARM  ) {)}

end
    [0030, 0005] (0009, END_SCP   ) {end}

show("array antes de ser ordenado")
    [0031, 0005] (0023, INS_SHOW  ) {show}
    [0031, 0009] (0010, BEGIN_PARM) {(}
    [0031, 0010] (0020, CONST_STR ) {"array antes de ser ordenado"}
    [0031, 0039] (0011, END_PARM  ) {)}

for i = 0, size - 2, :
    [0032, 0005] (0027, INS_FOR   ) {for}
    [0032, 0009] (0002, ID        ) {i}
    [0032, 0011] (0030, OP_ATRIB  ) {=}
    [0032, 0013] (0017, CONST_INT ) {0}
    [0032, 0014] (0016, SEPARATOR ) {,}
    [0032, 0016] (0002, ID        ) {size}
```

```
    [0032, 0021] (0040, OP_UNNEG  ) {-}
    [0032, 0023] (0017, CONST_INT ) {2}
    [0032, 0024] (0016, SEPARATOR ) {,}
    [0032, 0026] (0008, BEGIN_SCP ) {:}

    show(arr[i] & ", "
    [0033, 0009] (0023, INS_SHOW  ) {show}
    [0033, 0013] (0010, BEGIN_PARM) {(}
    [0033, 0014] (0002, ID        ) {arr}
    [0033, 0017] (0012, BEGIN_ARR ) {[}
    [0033, 0018] (0002, ID        ) {i}
    [0033, 0019] (0013, END_ARR   ) {]}
    [0033, 0021] (0039, OP_CONCAT ) {&}
    [0033, 0023] (0020, CONST_STR ) {", "}

end
    [0034, 0005] (0009, END_SCP   ) {end}

shellsort(arr,size)
    [0035, 0005] (0002, ID        ) {shellsort}
    [0035, 0014] (0010, BEGIN_PARM) {(}
    [0035, 0015] (0002, ID        ) {arr}
    [0035, 0018] (0016, SEPARATOR ) {,}
    [0035, 0019] (0002, ID        ) {size}
    [0035, 0023] (0011, END_PARM  ) {)}

show("array apos ser ordenado")
    [0036, 0005] (0023, INS_SHOW  ) {show}
    [0036, 0009] (0010, BEGIN_PARM) {(}
    [0036, 0010] (0020, CONST_STR ) {"array apos ser ordenado"}
    [0036, 0035] (0011, END_PARM  ) {)}

for i = 0, size - 2, :
    [0037, 0005] (0027, INS_FOR   ) {for}
    [0037, 0009] (0002, ID        ) {i}
    [0037, 0011] (0030, OP_ATRIB  ) {=}
    [0037, 0013] (0017, CONST_INT ) {0}
    [0037, 0014] (0016, SEPARATOR ) {,}
    [0037, 0016] (0002, ID        ) {size}
    [0037, 0021] (0040, OP_UNNEG  ) {-}
    [0037, 0023] (0017, CONST_INT ) {2}
    [0037, 0024] (0016, SEPARATOR ) {,}
    [0037, 0026] (0008, BEGIN_SCP ) {:}
```

```
        show(arr[i] & ", ")
        [0038, 0009] (0023, INS_SHOW  ) {show}
        [0038, 0013] (0010, BEGIN_PARM) {(}
        [0038, 0014] (0002, ID        ) {arr}
        [0038, 0017] (0012, BEGIN_ARR ) {[}
        [0038, 0018] (0002, ID        ) {i}
        [0038, 0019] (0013, END_ARR   ) {]}
        [0038, 0021] (0039, OP_CONCAT ) {&}
        [0038, 0023] (0020, CONST_STR ) {", "}
        [0038, 0027] (0011, END_PARM  ) {)}

    end
        [0039, 0005] (0009, END_SCP   ) {end}

    show(arr[size - 1])
        [0040, 0005] (0023, INS_SHOW  ) {show}
        [0040, 0009] (0010, BEGIN_PARM) {(}
        [0040, 0010] (0002, ID        ) {arr}
        [0040, 0013] (0012, BEGIN_ARR ) {[}
        [0040, 0014] (0002, ID        ) {size}
        [0040, 0019] (0040, OP_UNNEG  ) {-}
        [0040, 0021] (0017, CONST_INT ) {1}
        [0040, 0022] (0013, END_ARR   ) {]}
        [0040, 0023] (0011, END_PARM  ) {)}

end
        [0041, 0001] (0009, END_SCP   ) {end}
```

Hello World:

```
@ Código de teste

func void main():
        [0002, 0001] (0015, FUNC      ) {func}
        [0002, 0006] (0041, VOID      ) {void}
        [0002, 0011] (0001, MAIN      ) {main}
        [0002, 0015] (0010, BEGIN_PARM) {(}
        [0002, 0016] (0011, END_PARM  ) {)}
        [0002, 0017] (0008, BEGIN_SCP ) {:}
```

```
    show("Hello world!")
        [0003, 0005] (0023, INS_SHOW  ) {show}
        [0003, 0009] (0010, BEGIN_PARM) {(}
        [0003, 0010] (0020, CONST_STR ) {"Hello world!"}
        [0003, 0024] (0011, END_PARM  ) {)}

end
        [0004, 0001] (0009, END_SCP   ) {end}
```